Architecture for SuperSpeed data communication for USB 3.0 device using FPGA

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769 008, Odisha, India

Amit Kumar Amod

Architecture for SuperSpeed data communication for USB 3.0 device using FPGA

A thesis submitted in partial fulfilment

of the requirements for the degree of

Master of Technology

in Electronics and Communication Engineering

by

Amit Kumar Amod

Roll No. 211EC2074

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769 008, Odisha, India

May 2013

Architecture for SuperSpeed data communication for USB 3.0 device using FPGA

A Technical Report submitted in partial fulfilment

of the requirements for the degree of

Master of Technology

in

Electronics and Communication Engineering

by

Amit Kumar Amod

Roll No. 211EC2074

under supervision of

Dr. Debiprasad Priyabrata Acharya

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769 008, Odisha, India

May 2013

To my late grandmother

Department of Electronics and Communication Engineering
National Institute of Technology
Rourkela - 769 008, Odisha, India.

Certificate

This is to certify that the work done for the direction of thesis entitled **"Architecture for SuperSpeed data communication for USB 3.0 device using FPGA"** submitted by ***Mr. Amit Kumar Amod (Roll No. 211EC2074)*** in partial fulfilment of the requirements for the award of Master of Technology Degree in Electronics and Communication Engineering with specialization in VLSI Design and Embedded Systems at National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Place: NIT Rourkela

Date: 24th May, 2013

Dr. Debiprasad Priyabrata Acharya

Associate Professor

Acknowledgement

My first thanks are to the Almighty God, without whose blessings, I wouldn't have been writing this thesis.

I then would like to express my heartfelt thanks to my guide, **Dr. Debiprasad Priyabrata Acharya**, for his guidance, support, and encouragement during the course of my master study at the National Institute of Technology, Rourkela. I am especially indebted to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. Without his endless efforts, knowledge, patience and answers to my numerous questions, this research would have never been possible. The experimental methods and results presented in this thesis have been influenced by him in one way or the other. It has been a great honour and pleasure for me to do research under supervision of Dr. Debiprasad Priyabrata Acharya.

I am very much indebted to **Prof. Sukadev Meher**, Head of the Department, Electronics and Communication Engineering, National Institute of Technology, Rourkela for his support during my work.

I thank all the members of the Department of Electronics and Communication Engineering, and the Institute, who helped me by providing the necessary resources and in various other ways, in the completion of my work.

Finally, I thank my parents and my entire family member for their unlimited support and strength. Without their dedication and dependability, I could not have pursued my M. Tech. degree at the National Institute of Technology Rourkela.

Amit Kumar Amod

Abstract

The need for very large speed data communication leads to use of USB 3.0. This can be achieved by mixing the advantage of parallel and serial data transfer. This project work provides architecture for communication between USB 3.0 device controller (Cypress CYUSB3014) and USB 3.0 host controller (TUSB7320) at a data rate of 5.0 Gbps using Altera's Stratix IV (EP4SGX70DF29C3N) FPGA. To maintain synchronization between GPIF II and PCIe hard IP, two FIFO's are used. PLL is used to provide clock signal at various frequencies.

The physical layer provides signalling technology for SuperSpeed bus. The functionality of physical layer for USB 3.0 has been implemented in this project. Physical layer is functionally segregated in two parts, namely, transmitter and receiver. In transmitter module, the implementation of scrambler, 8b/10b encoder and parallel to serial converter is simulated using ModelSim-Altera 6.6d. And in receiver section, the implementation of serial to parallel converter, 8b/10b decoder and descrambling is similarly implemented. Both these modules are realized in Altera's Cyclone II (EP2C20F484C7) FPGA.

TABLE OF CONTENTS

List of figures

List of Tables

CHAPTER 1 Introduction

MOTIVATION

OBJECTIVE

ORGANISATION OF THIS THESIS

1 INTRODUCTION

1.1 MOTIVATION

The motivation for using Universal Serial Bus (USB) came from several considerations like ease-of-use, Port expansion etc. Initially, USB 1.0 provided two speeds, 12 Mbps and 1.5 Mbps, for peripherals. However, as PCs processed more data, users need a powerful protocol. To meet this requirement, USB 2.0 specification was published in 2000 to provide a third transfer rate of 480 Mb/s while retaining backward compatibility. New kinds of devices, large and inexpensive storage devices, and media formats are converging now days. These applications have prompted USB 3.0 [1], in 2008, which transfers data at 5 Gbps and is compatible with past protocols.

USB is the most powerful PC peripheral interconnect and has a great use in mobile and CE segments. The USB On-The-Go technique provides a way for two devices, capable of dual role, to be connected and decides which one is the "host". As new technologies emerges, new kind of devices, large inexpensive storage and new media format are emerging. This leads to the requirement of more bus bandwidth to maintain proper interaction. The USB is still the answer to connectivity for consumer electronics, PC and mobile architectures. It is a bidirectional, fast, dynamically attachable and low cost interface which fulfils the requirement of interconnection.

USB 3.0 is a cable bus supporting data exchange between a host computer and a wide range of simultaneously accessible peripherals. USB 3.0 utilizes dual bus architecture which provides its compatibility with USB 2.0. It provides both SuperSpeed and non-SuperSpeed connectivity.

1.2 OBJECTIVE

The main objective of this project is to use the SuperSpeed data rate in applications like imaging and video devices, printers, scanners etc. For that, the implementation is done on FPGA board [2]. Here, in this project work, the bridging between GPIF and PCI Express (Peripheral Component Interconnect Express) has been implemented on FPGA. The PCI Express Hard IP is used for the implementation [3] – [5]. And hence, according to the requirement we need to select the proper devices. Fig. 1.1 shows the block diagram of the project work.

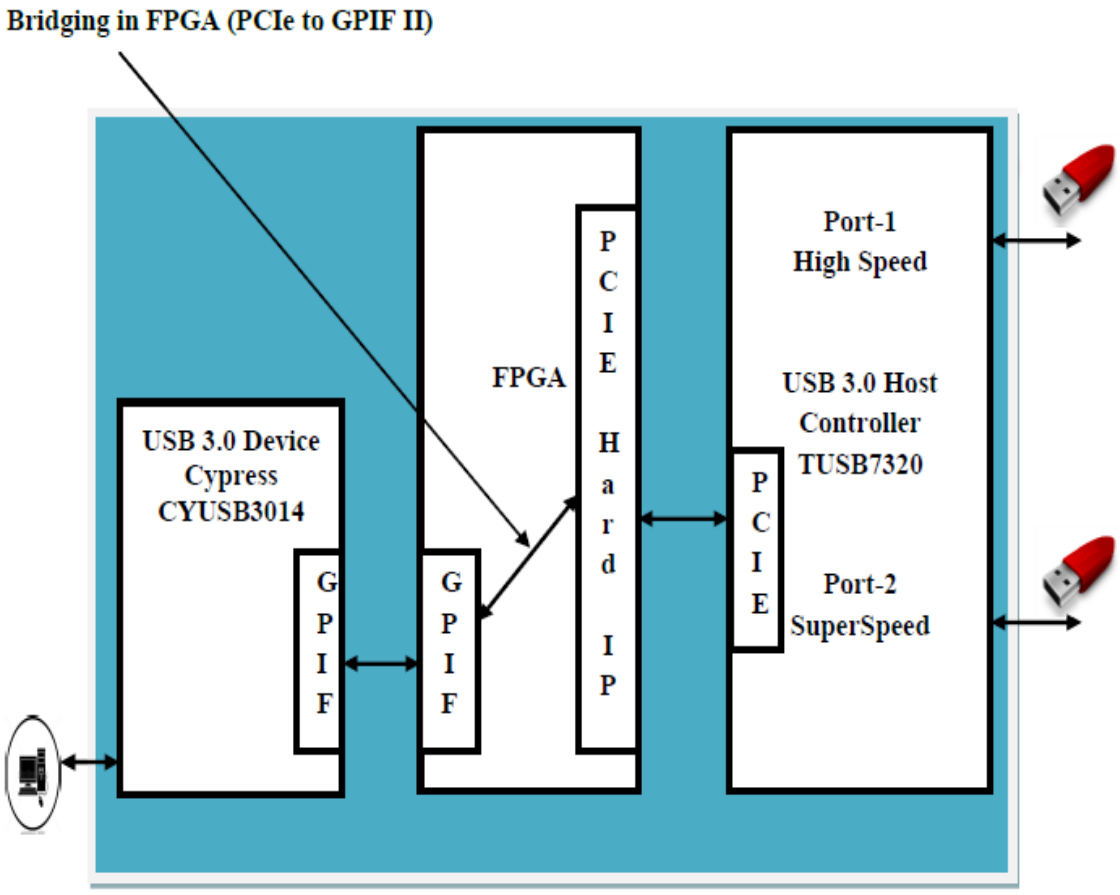

Fig. 1.1 Block diagram of Project work

The functionality of physical layer is shown in Fig. 1.2 and Fig. 1.3 representing transmitter and receiver block diagrams respectively.

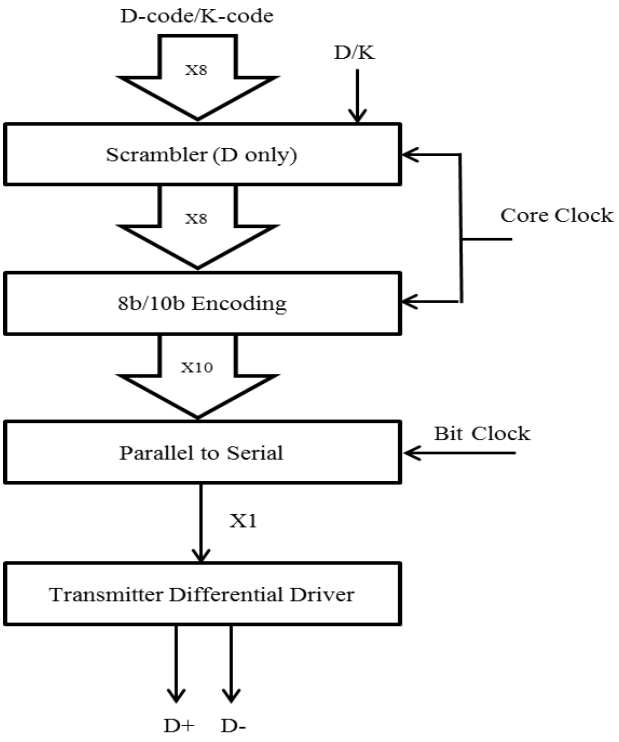

Fig. 1.2 Transmitter block diagram [6]

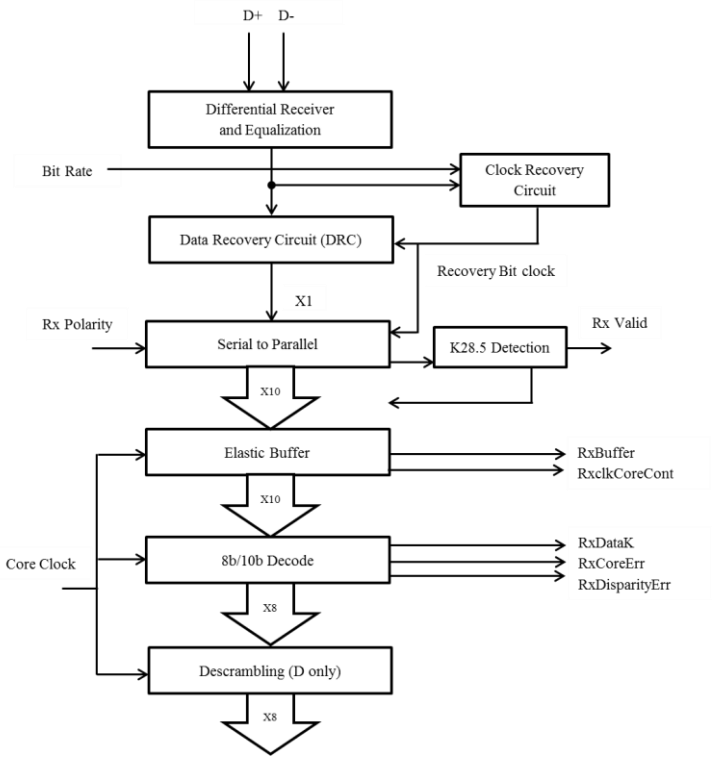

Fig. 1.3 Receiver block diagram [6]

The above functionality is implemented on Altera's FPGA. In this work, an input of 8-bit has to be given and the output will be of 1-bit in case of transmitter and vice-versa in

case of receiver. The goal of this research work was to develop a data acquisition solution to transfer data from a device controller to host controller and vice versa. Depending on the application, a large amount of data may need to be moved very quickly from the device controller to host controller.

1.3 ORGANISATION OF THIS THESIS

The thesis on Architecture for SuperSpeed data communication for USB 3.0 device using FPGA provides the detailed architecture for data transfer between device control and host control at a rate of 5 Gbps.

The basic components of a super speed data communication for USB3.0 are described in chapter 2. It includes Altera's Stratix IV FPGA, PLL, FIFO, Texas instrument's TUSB7320 and Cypress Device (CYUSB3014). For physical layer implementation, it includes linear feedback shift register, data scrambler and descrambler, and 8B/10B encoder and decoder.

Chapter 3 focuses on proposed architecture for data transfer between host and device controller. It consists of proposed architecture and simulation results of the components used in the proposed architecture like FIFO, GPIF and PLL etc.

Chapter 4 describes the implementation of physical layer components like linear feedback shift registers scrambler and descrambler, 8b/10b encoder and decoder and its simulation results.

CHAPTER 2 Component description

UNIVERSAL SERIAL BUS

FPGA

PLL

FIFO

TUSB7320

CYUSB3014

LINEAR FEEDBACK SHIFT REGISTER

DATA SCRAMBLING

8B/10B CODING

2 COMPONENT DESCRIPTION

2.1 UNIVERSAL SERIAL BUS (USB)

Universal Serial Bus (USB) is an industry standard developed that defines the cables and connectors used in a bus for connection between computers and electronic devices. It also defines protocols i.e. set of rules, which is used for communication and power supply.

2.1.1 USB 1.0

USB released its first version, USB 1.0 in January 1996. Its specified data rates were of 1.5 Mbps (Low-Bandwidth) and 12 Mbps (Full-Bandwidth), which are now called as Low-Speed and Full-Speed respectively. Due to timing and power limitations, it did not allow for extension cables or pass-through monitors.

2.1.2 USB 2.0

USB released its version USB 2.0 in April 2000 with added higher maximum signalling rate of 480 Mbps (effective throughput up to 35 Mbps) which is now called as Hi-Speed.

2.1.3 USB 3.0

In November 2008, USB released its version USB 3.0. The standard defines a new SuperSpeed mode with a signalling speed of 5 Gbps and a usable data rate of up to 4 Gbps. USB 3.0 reduces the time required for data transmission and hence reducing power consumption, and also it is backwards compatible with USB 2.0.

2.1.3.1 USB 3.0 ARCHITECTURE

USB 3.0 utilizes a dual-bus architecture which provides backward compatibility with USB 2.0. It provides both SuperSpeed and non-SuperSpeed signalling rate. It is similar to

USB2.0 that it is a cable bus supporting data exchange between a host computer and wide range of interconnected devices. All the interconnected devices uses host scheduled protocol i.e. bus allow devices to be attached, configured, used and detached while other devices are in operation.

USB 3.0 is a physical SuperSpeed bus combined in parallel with a physical USB 2.0 bus as shown in the Fig. 2.1.

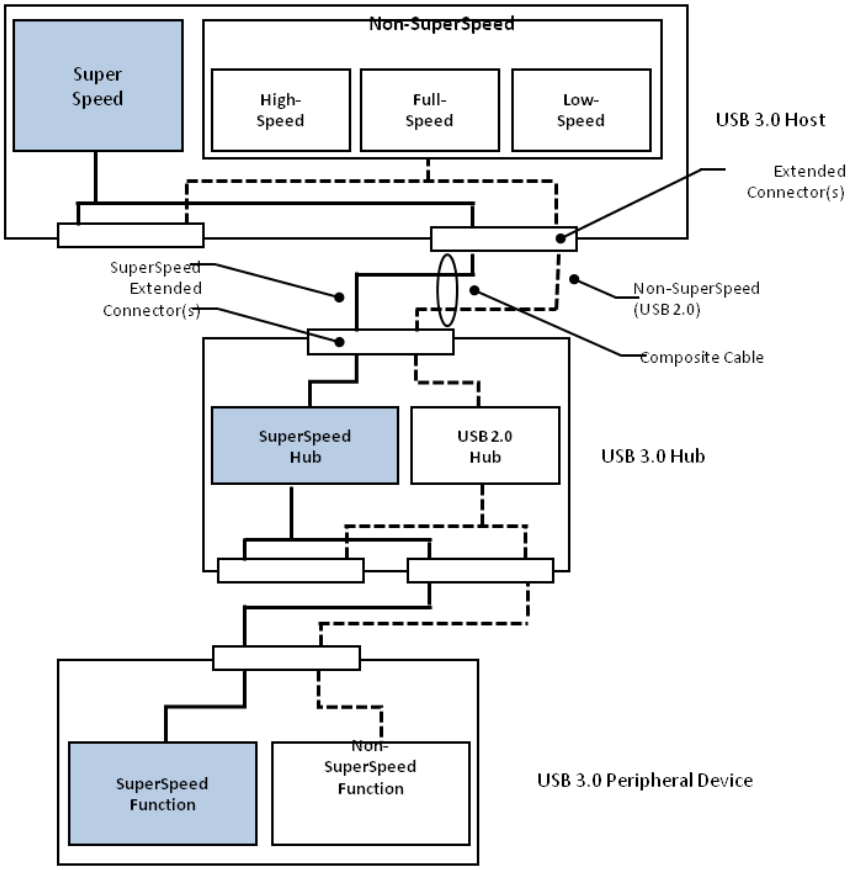

Fig. 2.1 USB 3.0 Dual Bus Architecture [6]

The USB 3.0 dual-bus architecture consists of the following components:

1. USB 3.0 interconnect

2. USB 3.0 devices

3. USB 3.0 host

The USB 3.0 interconnect defines the manner on which devices are connected and communicate with the host over the SuperSpeed bus. It includes topology, communication

layer and how they interacted to accomplish the data exchange. The USB 3.0 devices are sources or sink of information exchanges. They implemented the SuperSpeed communication layers and required device end to accomplish data exchange between drive on the host and logical functions of the devices. The USB 3.0 Host is a source or sink of information. It implements the SuperSpeed communication layers and required host-end to accomplish information exchange over the bus. It holds the SuperSpeed data activity schedule and management of the SuperSpeed bus and all the devices connected to it.

Fig. 2.2 illustrates a reference diagram of the SuperSpeed interconnect represented as communication layers through a topology oh host, zero to five levels of hub and devices.

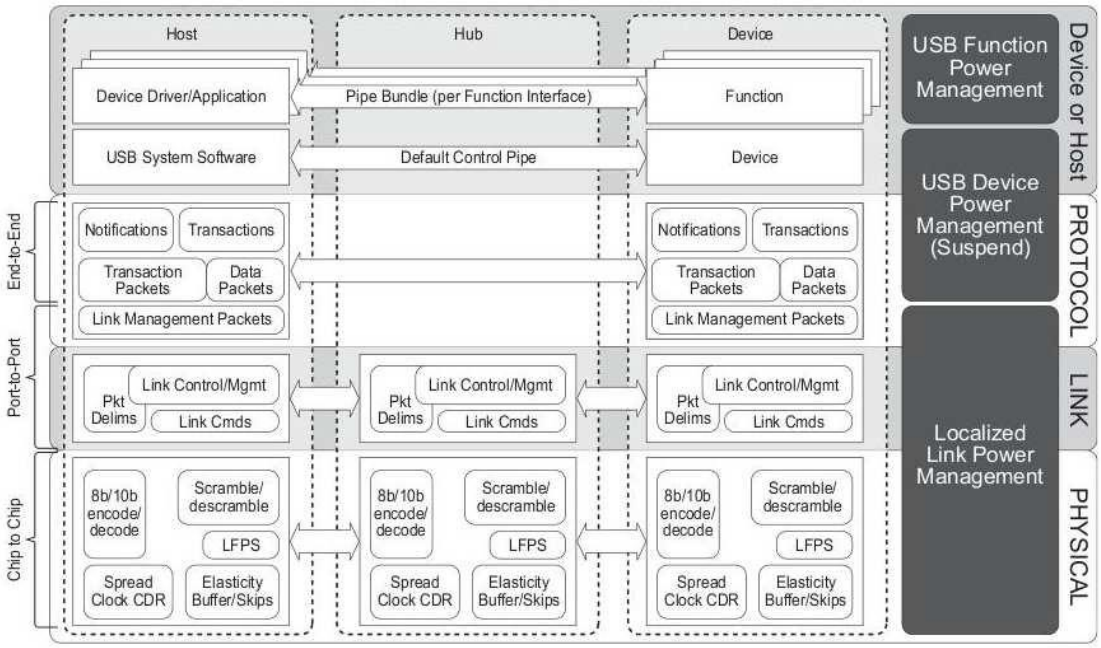

Fig. 2.2 SuperSpeed bus communication layer and Power Management Elements [6]

The rows (devices or host, protocol, link, physical) realize the communications layers of the SuperSpeed interconnect.

2.1.3.1.1 PHYSICAL LAYER

The physical layer [7] defines the PHY portion of a port and physical connection between a downstream facing port and upstream facing port of the device. SuperSpeed

physical connection is the comprised of two differential data pairs transmit and receive path. The nominal data rate is 5Gbps.

Electrical aspects of each path are characterized as a transmitter, channel and receiver. These are unidirectional differential link. Each differential link is AC-coupled with the capacitors located on the transmitter side. The channels hold electrical characteristics of the cable and connectors. The each differential link is initialized with receiver terminations. The Transmitter is responsible for detecting Receiver at other end of the same link. When receiver termination occurs but no signalling is occurring then it is consider to be an electrical idle state and in this state, low frequency periodic signal is generated which is simple to generate and consume very little power.

Each PHY has its own clock domain. The USB 3.0 cable does not have a reference clock so bit level timing synchronization relies on the local receiver to align bit recovery clock to the remote transmitter's clock by phase-locking to the signal transitions in the received bit stream. To ensure the proper transitions occur in the bit stream independent of the data content being transmitted, the transmitter encode the data and special characters using 8b/10b code. Control symbols are used to achieve byte alignment and are used for framing data and managing the link.

2.1.3.1.2 LINK LAYER

It is logical and physical connection of the two ports. The connected ports are called link partners. A Port has logical portion and physical portion, link layer [7] defines the logical portion.

Logical portion include:

1. Initialization of physical layer and event management that is, removal, connect, and power management.
2. State machines and buffering for managing information exchanges. They implements

protocol for reliable delivery of packet headers, flow control, and link power management.

3. Buffering for data and protocol layer information elements.
4. Detect receive packets and error checks for received header packets.
5. Provide an appropriate interface to the protocol layer for information exchanges.

Physical portion include:

1. Managing state of its PHY that is, power management and events.
2. Transmit and receive byte streams.

2.1.3.1.3 PROTOCOL LAYER

It defines end to end communication rules between a host and device. It provides data information exchange between host and device endpoint. This communication relationship is called PIPE [8]. Host determines when application data is transferred between host and device.

Device is able to asynchronously request service from the host. Protocol communications are accomplished via the exchange of packets. These packets are sequence of data with specific control sequence.

Packet headers are the building block of protocol layer. Packet headers are fixed in size packets with a type and a subtype. A small record within a packet header is utilized by the link layer to manage the flow of the packet from port to port. Packet headers are passed through the link layer reliably. The remaining fields are utilized by the end –to –end protocol. Application data is transmitted inside the data packet payloads. They are encoded with data packet headers.

2.1.3.1.4 DEVICES

All SuperSpeed devices share their base architecture with that of USB 2.0. They are required to carry information for configuration and identification. All devices support one or more pipe through which the host will communicate with the device. All devices must support assigned pipe at endpoint zero to which device's default control pipe is attached.

2.1.3.2 ROBUSTNESS AND ERROR HANDLING

For the robustness and error handling, there are several attributes

1. CRC protection for header and data packets.
2. Link level header packets to ensure reliable delivery.
3. Detection of attach and detach and system level configuration of resource.
4. Data and control pipe constructs for ensuring independent interactions between functions.

To provide protection against the bit error each packet includes CRC. The protocol includes separate CRC for header and data packet payloads. Link control word has its own CRC. A failed CRC in the header and link control word consider being a serious error. In such situation link level retry to recover from error. Link layer and physical layer works together to provide reliable packet header transmission. The Physical layer provides error rate and link layer provide error checking. The only way to recover error in hardware is to retry the header packet.

2.1.3.3 POWER MANAGEMENT

SuperSpeed provides power management in bus architecture, link, device and function. These areas are in coupled with each other based on allowable power state transitions. Link power management occur asynchronously on every link in the connected hierarchy. It depends upon device, host, or combination of both. The Link power state may be

driven by host or downstream port. Link power states are propagated upward by hubs. The decisions to change link power state are made locally. Additionally Links that are not being used can be placed in low power state.

The host does not directly control visibility of individual link power state that means one or more link in the path between host and device can be in reduced power state. There are in-band protocol mechanisms that force this link to transition to the operation power state and notify the host that a transition has occurred. A device initiating a communication on the bus with its upstream link in a reduced power state will first transition its link into an operational state. This will cause all links between it and host to transition to the operation state.

The key points of link power management includes

1. Devices send asynchronous ready notification to the host.
2. Packets are routed, allowing links that are not involved in communication to transition into or to remain in low power state.
3. Packets that encounter ports in low power state cause those ports to transition out of low power state.

SuperSpeed provides function power management in addition to device power management. For multi functions devices each function can be placed in low power state. The device will transition into the suspended state by the host via a port command. The devices will not automatically transition into the suspended state when all the individual functions within it are suspended.

2.2 FPGA

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing and hence it is called as field-programmable. Generally, FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated

circuit (ASIC). Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast IOs and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. Floor planning enables resources allocation within FPGA to meet these time constraints. FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non-recurring engineering costs more relative to an ASIC design, offer advantages for many applications.

FPGAs contain programmable logic components called logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together, somewhat like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

Some FPGAs have analog features in addition to digital functions. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unacceptably and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slow. Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signalling channels. A few mixed signal FPGAs have integrated peripheral analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip [9]. Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric and field-programmable

analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

2.2.1 STRATIX IV FPGA

Stratix IV FPGAs [10] provide the highest density, highest performance and lowest power of any 40-nm FPGA. With enhanced (E) and enhanced with transceivers (GX and GT) variants, Stratix IV FPGAs address many applications, such as wireless and wireline communications, military and broadcast. This high-performance 40-nm FPGA family includes the best-in-class 11.3-Gbps transceivers.

The Stratix IV GX FPGA is fully PCI-SIG compliant for PCI Express Gen1 and Gen2 (x1, x4 and x8) and is on the PCI-SIG integrators list. Altera's STRATIX IV FPGA consists of Transceivers and Hard IP blocks for PCIe.

2.2.1.1 PCI EXPRESS

Peripheral Component Interconnect (PCI) Express [11] – [14] is a high-performance interconnect protocol used in a variety of applications including network adapters, storage area networks, embedded controllers, graphic accelerator boards and audio-video products. The PCI Express protocol is software backwards-compatible with the earlier PCI and PCI-X protocols, but is significantly different from its predecessors. It is a packet-based, serial, point-to-point interconnect between two devices. The performance is scalable based on the number of lanes and the generation that is implemented. Altera offers both endpoints and root ports that are compliant with PCI Express Base Specification 1.0a or 1.1 for Gen1 and PCI Express Base Specification 2.0 for Gen1 or Gen2. Both endpoints and root ports can be implemented as a configurable hard IP block rather than programmable logic, saving significant FPGA resources. The IP Compiler for PCI Express is available in ×1, ×2, ×4 and ×8 configurations.

2.2.1.2 PCIE HARD IP

The PCIE Express hard intellectual property (PCIE HARD IP) [15] block embeds the PCI Express protocol stack into the Altera's FPGA. The hard IP block includes the transceiver modules, physical layer, data link layer and transaction layer. For Stratix IV (GT and GX) FPGAs, the hard IP block targets PCI Express Base Specification Rev. 2.0 and 1.1. **Fig. 2.3** shows block diagram of a PCI Express hard IP block.

The following are the advantages of using PCIE Hard IP:

1. Resource saving of 8K to 30K logic elements (LEs) per hard IP instance, depending on the initial core configuration mode.
2. Embedded memory buffers included in the hard IP.
3. Pre-verified, protocol-compliant complex IP.
4. Shorter design and compile times with timing closed block.
5. Substantial power savings relative to a soft IP core with equivalent functionality.

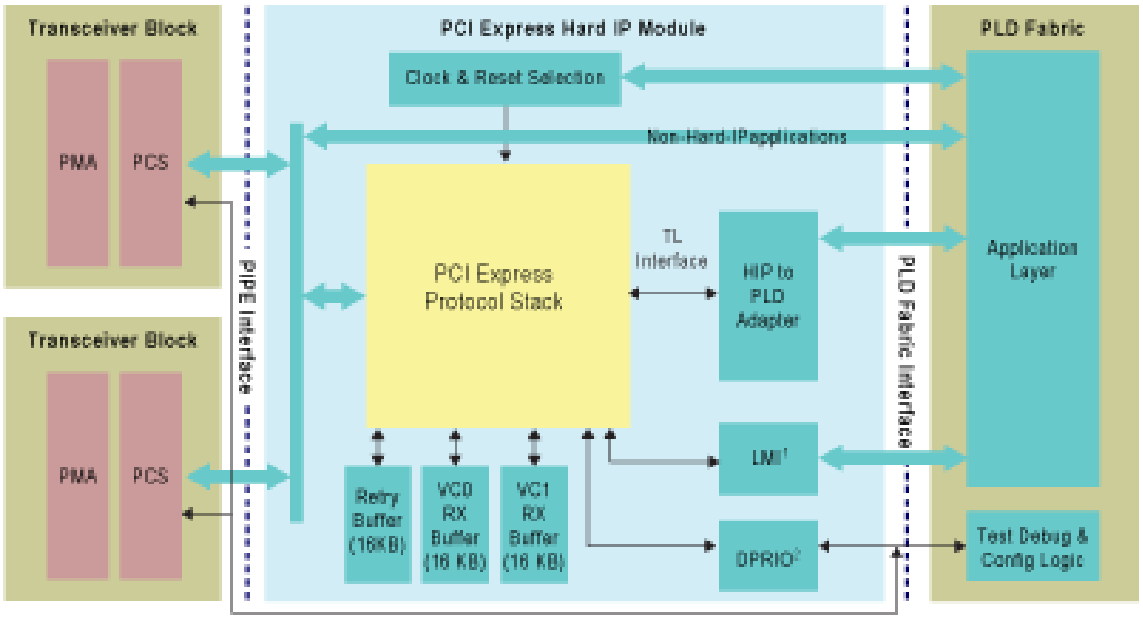

Fig. 2.3 PCIe Hard IP block [15]

2.2.1.3 TRANSCEIVER

Stratix IV transceivers [16] include dedicated circuitry to implement standard and proprietary protocols operating between 600 Mbps and 8.5 Gbps in Stratix IV variant. The transceivers are also capable of supporting legacy protocols and protocols with multiple data rates. When augmented with Altera IP, Stratix IV GT FPGA and Stratix IV GX FPGA transceivers provide a complete, low-risk solution for serial protocol implementation. Fig. 2.4 shows block diagram of STRATIX IV GX Transceivers, PMA and PCS.

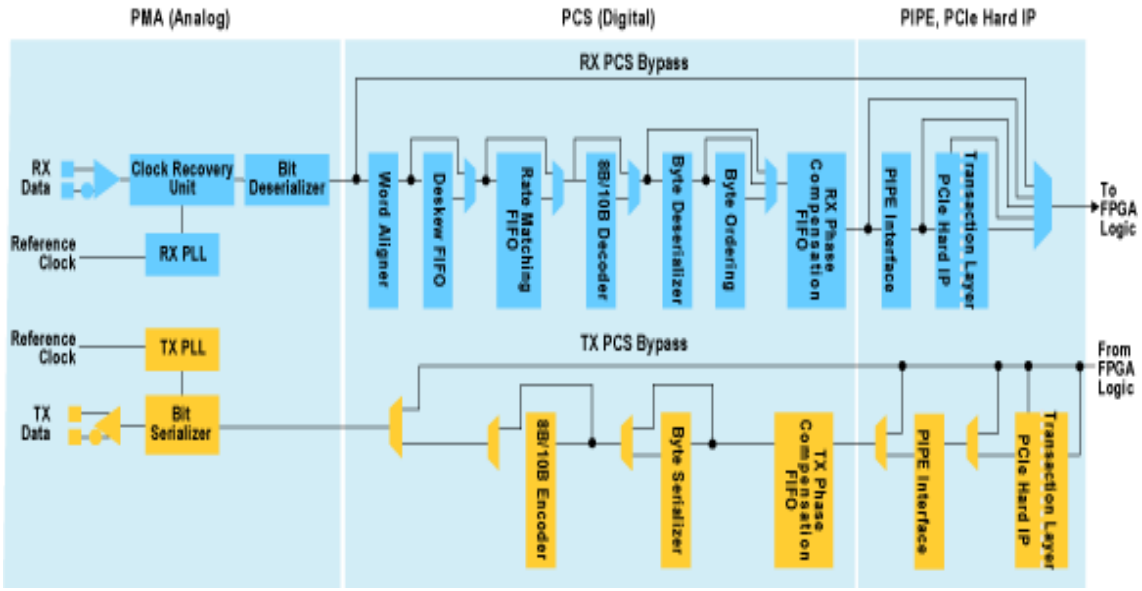

Fig. 2.4 STRATIX IV GX Transceivers, PMA and PCS Block Diagram [16]

2.3 PLL

The Phase-Locked Loop (PLL) is a closed-loop frequency-control system that compares the phase difference between the input signal and the output signal of a voltage-controlled oscillator (VCO). The negative feedback loop of the system forces the PLL to be phase-locked.

PLLs are widely used in telecommunications, computers, and other electronic applications. The PLL can be used to generate stable frequencies, recover signals from a

noisy communication channel, or distribute clock signals throughout your design. The PLL consists of a pre-divider counter (N counter), a phase-frequency detector (PFD) circuit, a charge pump, loop filter, a VCO, a feedback multiplier counter (M counter), and post-divider counters (K and V counters).

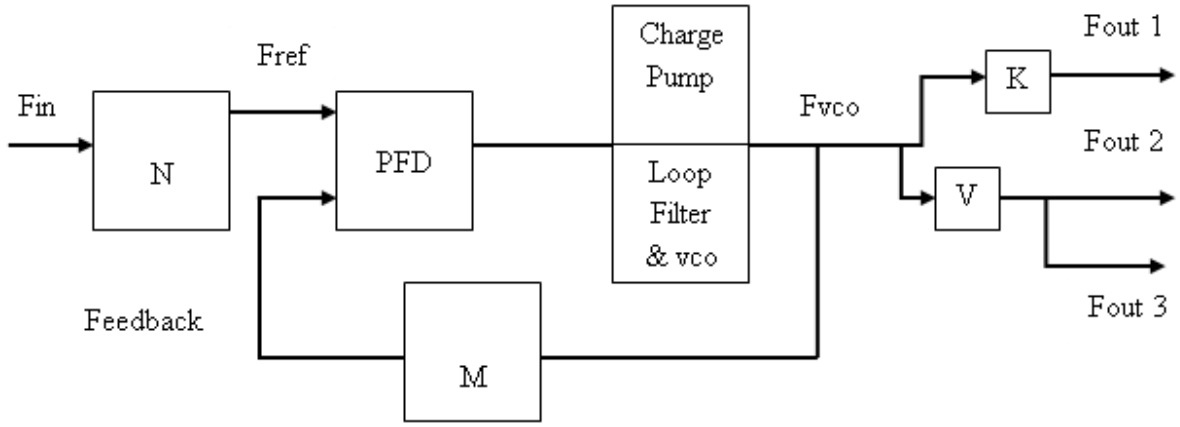

Fig. 2.5 PLL block diagram

The PFD detects the differences in phase and frequency between its reference signal (Fref) and feedback signal (Feedback), controls the charge pump, and controls a loop filter that converts the phase difference to a control voltage. This voltage controls the VCO. Based on the control voltage, the VCO oscillates at a higher or lower frequency, which affects the phase and frequency of the Feedback signal. After the Fref signal and the Feedback signal have the same phase and frequency, the PLL is said to be phase-locked.

Inserting the M counter in the feedback path causes the VCO to oscillate at a frequency that is M times the frequency of the Fref signal. The Fref signal is equal to the input clock (Fin) divided by the pre-scale counter (N). The reference frequency is described by the equation Fref = Fin/N. The VCO output frequency is Fvco = Fin × M/N, and the output frequency of the PLL is described by the equation Fout = (Fin × M)/(N × K) for the signals shown in Fig. 2.5.

2.3.1 ALTPLL

The Altera's Quartus II software provides the ALTPLL MegaWizard interface to specify the PLL circuitry in the supported devices. One can use the MegaWizard Plug-In Manager to configure the ALTPLL MegaWizard interface and build ALTPLL megafunctions efficiently. The ALTPLL MegaWizard interface appears in the I/O category in the MegaWizard Plug-In Manager. The types of PLL supported by the megafunction depend on the device family. Device families typically support one or two PLL types. The Stratix series supports two types of PLLs. The two PLL types supported within a device family are identical in their analog portions and differing slightly in the digital portion.

2.3.1.1 OPERATING MODES

The ALTPLL megafunction supports up to five different clock feedback modes, depending on the selected device family. Each mode allows clock multiplication and division, phase shifting, and duty-cycle programming.

The following list describes the operation modes for the ALTPLL megafunction:

1. **Normal mode:** The PLL feedback path source is a global or regional clock network, minimizing clock delay to registers for that clock type and specific PLL output.
2. **Source-Synchronous mode:** The data and clock signals arrive at the same time at the input pins. In this mode, the signals are guaranteed to have the same phase relationship at the clock and data ports of any Input Output Enable (IOE) register.
3. **Zero-Delay Buffer mode:** The PLL feedback path is confined to the dedicated PLL external output pin. The clock port driven off-chip is phase aligned with the clock input for a minimal delay between the clock input and the external clock output.
4. **No Compensation mode:** The PLL feedback path is confined to the PLL loop. It has no clock network or other external source. A PLL in no-compensation mode has no clock network compensation, but clock jitter is minimized.

5. **External Feedback mode:** The PLL compensates for the feedback input to the PLL. The delay between the input clock pin and the feedback clock pin is minimized.

2.4 FIFO

FIFO is an acronym for First In – First Out, which is an abstraction related to ways of organizing and manipulation of data relative to time and prioritization. FIFOs are used commonly in electronic circuits for buffering and flow control which is from hardware to software. In hardware form, a FIFO primarily consists of a set of read and write pointers, storage and control logic. Storage may be SRAM, flip-flops, latches or any other suitable form of storage. For FIFOs of non-trivial size, a dual-port SRAM is usually used where one port is used for writing and the other is used for reading.

A synchronous FIFO is a FIFO where the same clock is used for both reading and writing. An asynchronous FIFO uses different clocks for reading and writing. Asynchronous FIFOs introduce metastability issues. A common implementation of an asynchronous FIFO uses a Gray code or any unit distance code for the read and write pointers to ensure reliable flag generation. One further note concerning flag generation is that one must necessarily use pointer arithmetic to generate flags for asynchronous FIFO implementations. Conversely, one may use either leaky bucket approach or pointer arithmetic to generate flags in synchronous FIFO implementations. Examples of FIFO status flags include: full, empty, almost full, almost empty, etc.

2.4.1 ALTFIFO

Altera provides FIFO functions through the parameterizable single-clock FIFO (SCFIFO) and dual-clock FIFO (DCFIFO) megafunctions. The FIFO functions are mostly applied in data buffering applications that comply with the first-in-first-out data flow in

synchronous or asynchronous clock domains. The specific names of the megafunctions are as follows:

1. SCFIFO: single-clock FIFO
2. DCFIFO: dual-clock FIFO (supports same port widths for input and output data)
3. DCFIFO_MIXED_WIDTHS: dual-clock FIFO (supports different port widths for input and output data)

Fig. 2.6 show input and output ports of DCFIFO.

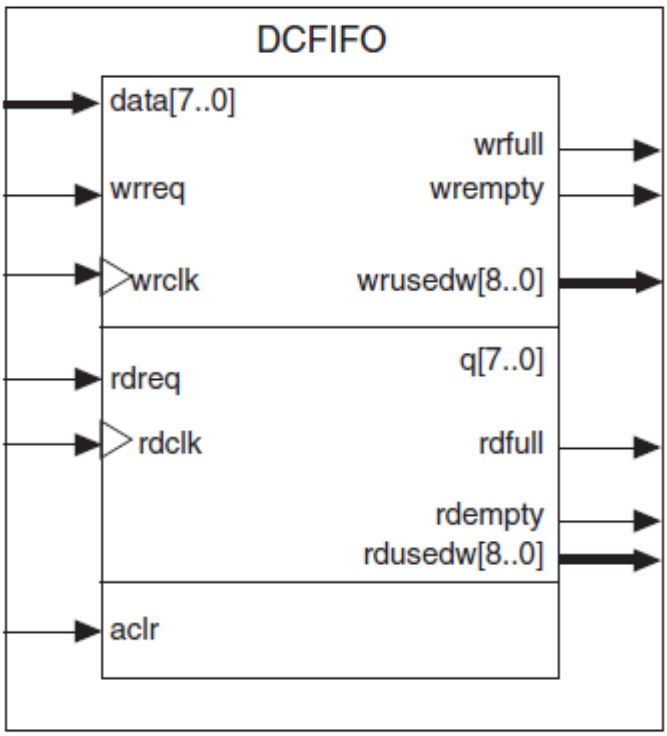

Fig. 2.6 INPUT AND OUTPUT PORTS OF DCFIFO

In my project work, DCFIFO_MIXED_WIDTH has been used because the PCIE hard IP supports 64-bit data width whereas the CYUSB3014 supports 32-bit data width. To provide up-conversion and down-conversion between 64-bit and 32 bit, the DCFIFO_MIXED_WIDTH is used.

2.5 TUSB7320

It is a Texas Instrument's device [17] which is a host controller. It is a USB 3.0 compatible device, which supports PCIe x1 Gen2 Interface with two downstream ports. Each downstream port may be independently enabled or disabled. Table 2-1 describes the input and output terminals towards PCI Express.

Table 2-1 PCI Express Signals

NAME	DIRECTION	DESCRIPTION
PCIE_TXP PCIE_TXN	O	PCI Express transmitter differential pair.
PCIE_RXP PCIE_RXN	I	PCI Express receiver differential pair.
WAKE	O	Wake. Wake is an active low signal that is driven low to reactivate the PCI Express link hierarchy's main power rails and reference clocks.
CLKREQ	O	PCI Express REFCLK Request signal.
PCIE_REFCLKP PCIE_REFCLKN	I	PCI Express Reference Clock. PCIE_REFCLKP and PCIE_REFCLKN comprise the differential input pair for the 100-MHz system reference clock.
PERST	I	PCI Express Reset Input. The PERST signal is used to signal when the system power is stable. The PERST signal is also used to generate an internal power on reset

2.6 CYUSB3014

It is a Cypress's device [18] which is a device controller. It provides highly integrated and flexible features which supports USB 3.0 functionality so that it can be added to any system. It has a fully configurable, parallel, General Programmable Interface called GPIF II, which can connect to any processor, ASIC, or FPGA. Fig. 2.7 shows Logic Block Diagram of CYUSB3014.

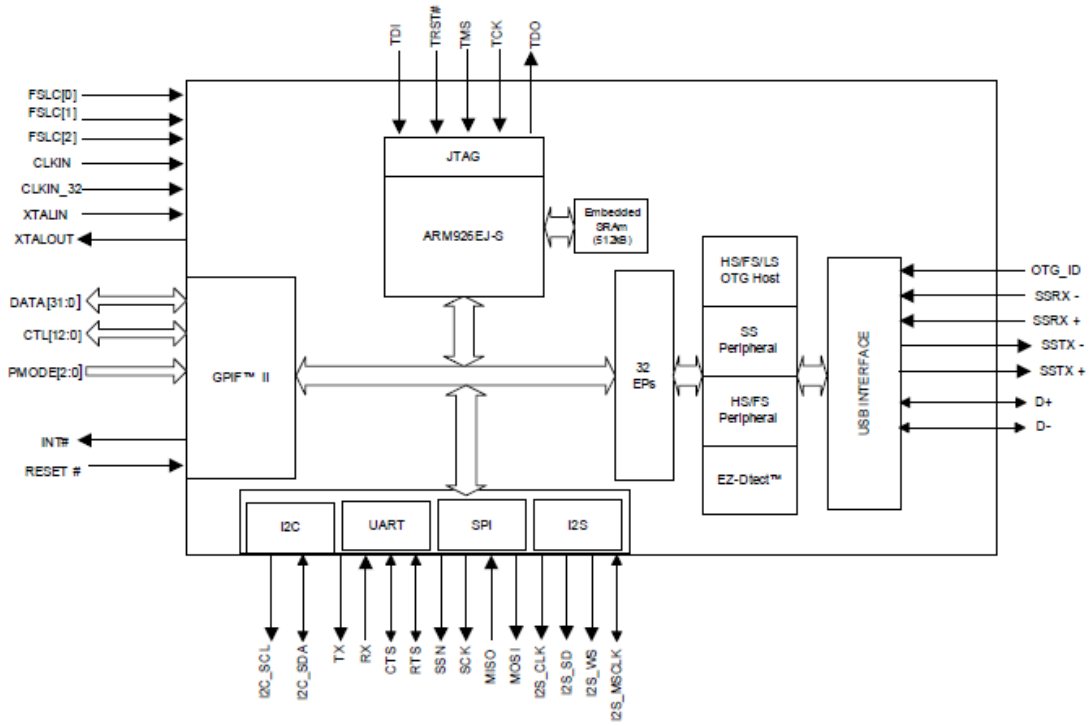

Fig. 2.7 Logic Block Diagram of CYUSB3014 [3]

2.6.1 GPIF II

It provides easy and glueless connectivity to popular interfaces such as asynchronous SRAM, synchronous and synchronous Address Data Multiplexed interface, parallel ATA, and so on. One popular implementation of GPIF II is the slave FIFO interface [19, 20]. This interface is used for applications in which the external device connected to FX3 accesses the

FX3 FIFOs, reading from or writing data to them. Direct register accesses are not performed over the slave FIFO interface. Fig. 2.8 shows Slave FIFO Interface diagram of GPIF II.

The GPIF II is a programmable state machine that enables a flexible interface that may function either as a master or slave in industry standard or proprietary interfaces. Both parallel and serial interfaces may be implemented with GPIF II.

The features of the GPIF II are summarized as follows:

1. Functions as master or slave.
2. Provides 256 firmware programmable states.
3. Supports 8 bit, 16 bit and 32 bit parallel data bus.
4. Enables interface frequencies up to 100 MHz.
5. Supports 14 configurable control pins when 32 bit data bus is used. All control pins can be either input/output or bidirectional.
6. Supports 16 configurable control pins when 16/8 data bus is used. All control pins can be either input/output or bidirectional.

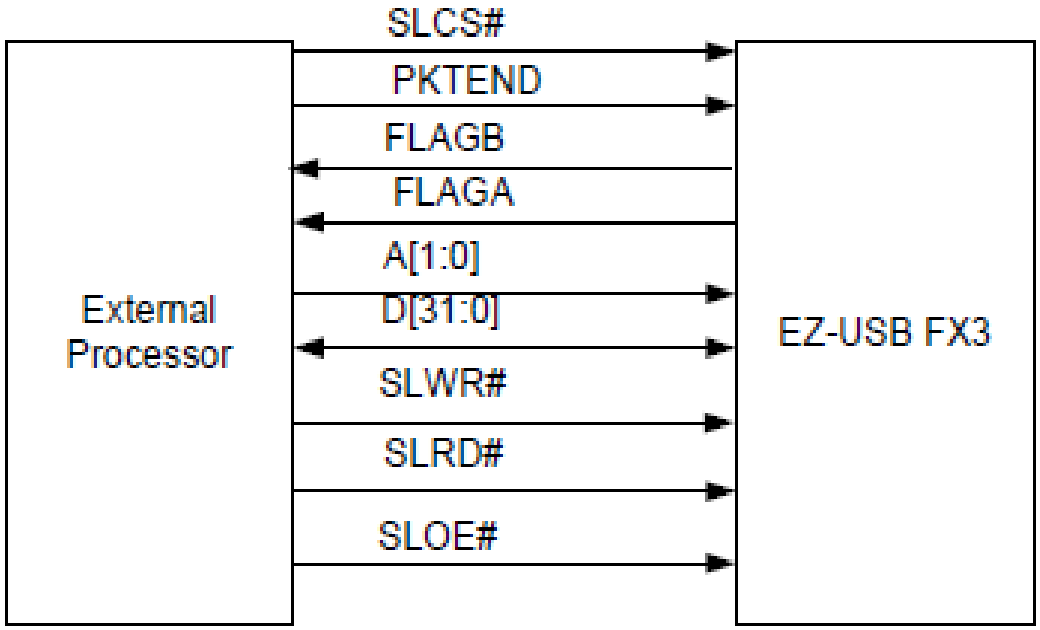

Fig. 2.8 Slave FIFO Interface [3]

GPIF II state transitions occur based on control input signals. The control output signals are driven as a result of GPIF II state transitions. The behaviour of the GPIFII state machine is defined by a GPIFII descriptor. The GPIF II descriptor is designed such that the required interface specifications are met. 8kB of memory (separate from the 512kB of embedded SRAM) is dedicated as GPIF II Waveform memory where the GPIF II descriptor is stored in a specific format.

(Linear feedback shift register)In computing, a linear feedback shift register (LFSR) [21] is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is XOR. Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value.

The initial value of the LFSR is called the seed and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. Fig. 2.9 shows a 4-bit LFSR.

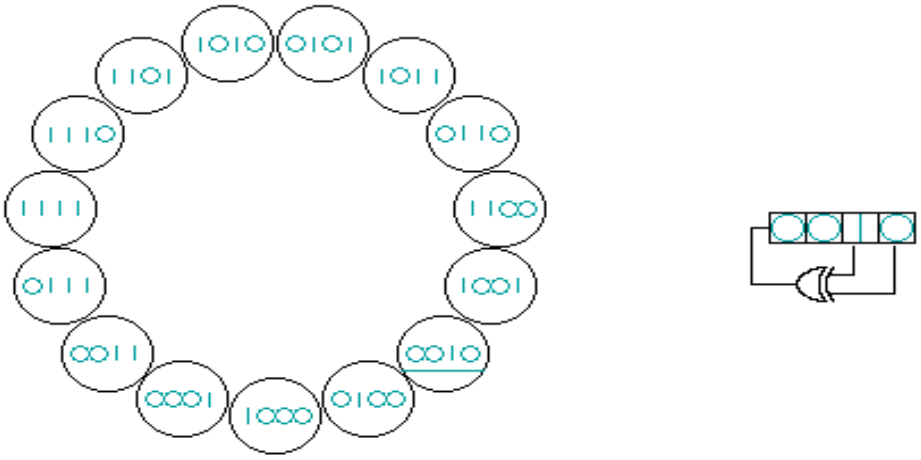

Fig. 2.9 A 4-bit LFSR [21]

2.7 DATA SCRAMBLING

The scrambling function is implemented using a free running Linear Feedback Shift Register (LFSR). On the transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the receiver side, descrambling is applied to characters after 8b/10b decoding. The LFSR is reset whenever a COM symbol is sent or received.

The data scrambling rules are as follows:

1. The LFSR implements the polynomial

   $$G(X) = X^{16} + X^5 + X^4 + X^3 + 1$$

2. The LFSR value shall be advanced eight serial shifts for each symbol except for SKP.
3. All 8b/10b D-code, except those within the Training Sequence Ordered Sets shall be scrambled.
4. K code shall not be scrambled.
5. The initialized value of an LFSR seed (D0-D15) shall be FFFFH. After COM leaves the transmitter LFSR, the LFSR on the transmit side shall be initialized. Every time COM enters the receive LFSR, the LFSR on the receive side shall be initialized.

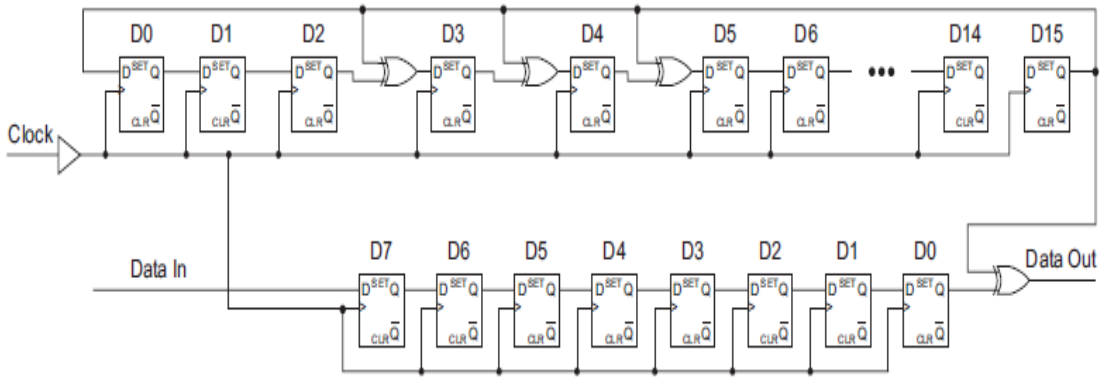

Fig. 2.10 LFSR with scrambling polynomial [21]

2.8 8B/10B SYMBOL CODING

In telecommunications, 8b/10b is a line code that maps 8-bit symbols to 10-bit symbols to achieve DC-balance and bounded disparity and yet provide enough state changes to allow reasonable clock recovery. This means that the difference between the count of 1s and 0s in a string of at least 20 bits is no more than 2 and that there are not more than five 1s or 0s in a row. This helps to reduce the demand for the lower bandwidth limit of the channel necessary to transfer the signal. An 8b/10b code can be implemented in various ways, where the design may focus on specific parameters such as hardware requirements, dc-balance etc.

As the scheme name suggests, 8 bits of data are transmitted as a 10-bit entity called a symbol, or character. The low 5 bits of data are encoded into a 6-bit group (the 5b/6b portion) and the top 3 bits are encoded into a 4-bit group (the 3b/4b portion). These code groups are concatenated together to form the 10-bit symbol that is transmitted on the wire. The data symbols are often referred to as D.x.y where x ranges over 0–31 and y over 0–7. Standards using the 8b/10b encoding also define up to 12 special symbols (or control characters) that can be sent in place of a data symbol. They are often used to indicate start-of-frame, end-of-frame, link idle, skip and similar link-level conditions. At least one of them

(i.e. a "comma" symbol) needs to be used to define the alignment of the 10 bit symbols. They are referred to as K.x.y and have different encodings from any of the D.x.y symbols.

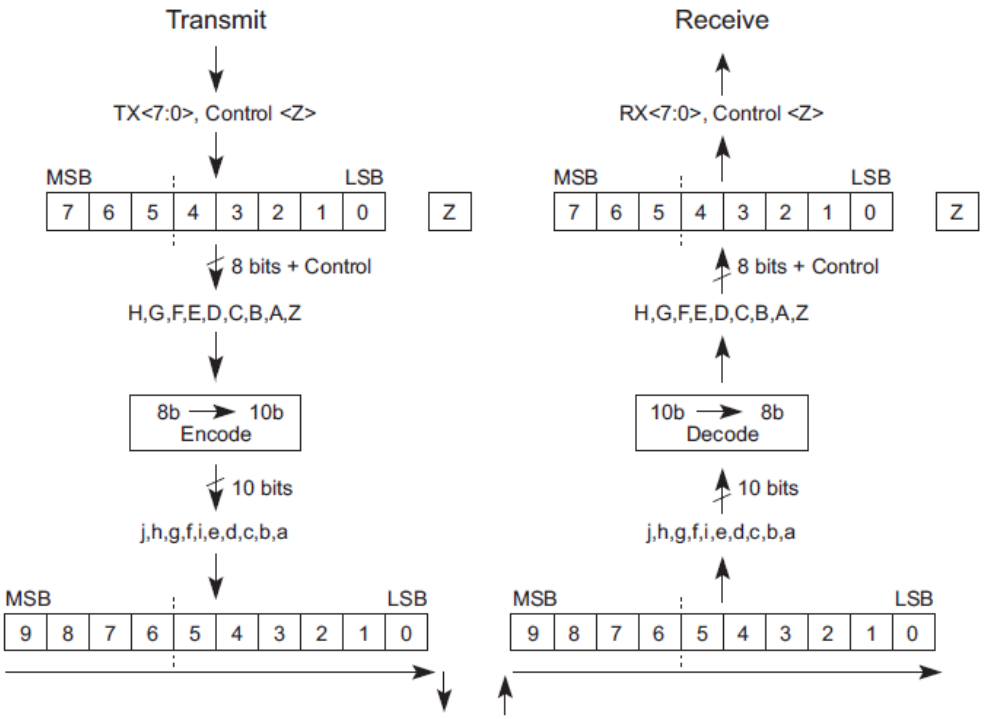

Fig. 2.11 8b/10b symbol coding

Because 8b/10b encoding uses 10-bit symbols to encode 8-bit words, some of the possible 1024 (10 bit, 210) codes can be excluded to grant a run-length limit of 5 consecutive equal bits and grant that the difference of the count of 0s and 1s is no more than 2. Some of the 256 possible 8-bit words can be encoded in two different ways. Using these alternative encodings, the scheme is able to affect long-term DC-balance in the serial data stream. This permits the data stream to be transmitted through a channel with a high-pass characteristic. SuperSpeed uses the 8b/10b transmission code. Fig. 2.11 describe the 8b/10b encoding and decoding process for the transmitter and receiver respectively.

CHAPTER 3 Proposed Architecture for Data Transfer between Host and Device Controller

PROPOSED DESIGN

SIMULATION RESULTS

3 PROPOSED ARCHITECTURE FOR DATA TRANSFER BETWEEN HOST AND DEVICE CONTROLLER

3.1 PROPOSED DESIGN

The data communication between the host controller and PCIe hard IP is a serial communication, whereas the output from PCIe hard IP is a 64-bit parallel data. The device controller supports maximum data width of 32-bit. Hence to communicate between device controller and PCIe hard IP, we need to convert 64-bit data to 32-bit and vice versa for duplex communication. The host controller and device controller can operate at difference frequencies. CYUSB3014 (device controller) and TUSB7320 (host controller) works at different frequency. Hence to maintain synchronisation between both the devices, we need to use two FIFO, one for transmission of data and another for reception of data. The FIFO used is a double clock mixed width FIFO. It converts 64-bit data to 32-bit data and vice versa at transmitter and receiver side respectively. PLL is used to provide clock signal at multiple frequencies. An external clock signal is given to PLL to produce output clock signal of different frequencies. Altera's STRATX IV FPGA is used to implement the architecture. It consists of PCIe hard IP block and transceiver blocks. Altera's ModelSim is used to simulate the code.

A differential pair of input is given to the hard IP from the host controller in terms of PCIE_TXP and PCIE_TXN. The hard IP will check first whether the FIFO controller is ready or not via rx_st_ready pin. If the FIFO controller is ready, then it will start transmitting data to FIFO_RX until the end of packet goes high. In this process, the start of packet will go high for one clock cycle and the valid pins go high. The FIFO_RX converts the 64-bit parallel data to 32-bit parallel data. The output data of FIFO_RX is given to tri state buffer. The wrusedw_rx tells whether the FIFO_RX is full or not.

A bidirectional data bus is used in the device controller so that data can be read or written depending on the usage. The device controller gives an interrupt signal to the GPIF controller. The GPIF controller checks whether there is data present in FIFO_RX or not. Depending on the content of FIFO_RX, the rdempty_rx signal changes its state. If there is data in FIFO_RX, then GPIF controller will select the device controller for write operation through a chip select signal.

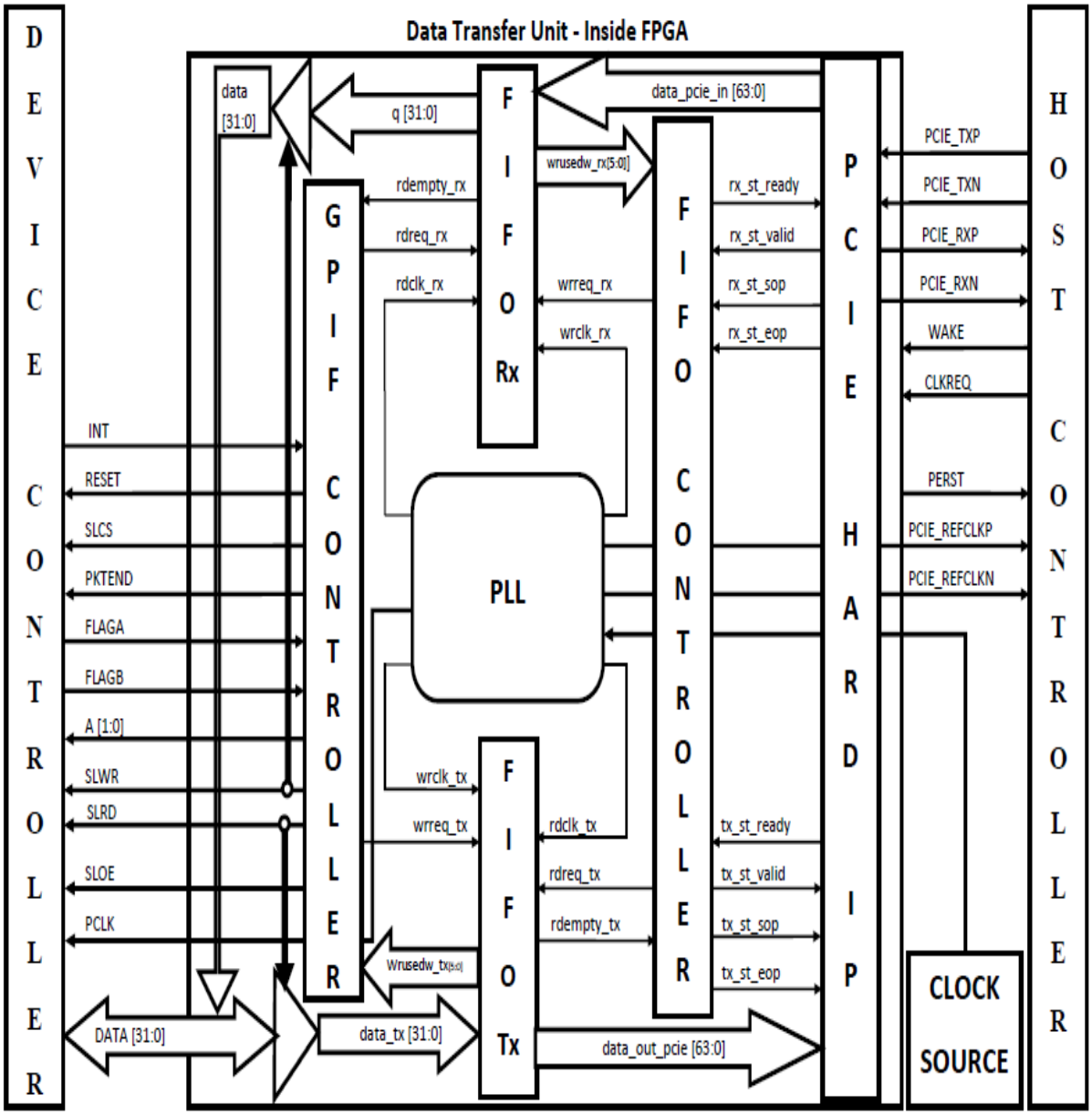

Fig. 3.1 Proposed architecture

The FLAGA and FLAGB show the state of FIFO present in the device controller. FLAGA represents the empty state and FLAGB represents the full state of the FIFO present in the device controller. If the FIFO in the device controller is empty, then GPIF controller starts sending the data from FIFO_RX via tri state buffer to the device controller till packet end occurs. If the FIFO in the device controller is full, then the GPIF controller reads data from the device controller and gives it to FIFO_TX.

If the FIFO_TX is full, then the write operation stops. This is determined by the status of wrused_tx signal. If data is available in FIFO_TX, then FIFO controller sends read request signal to FIFO_TX and data transmission starts only when hard IP is ready to accept data till end of packet. A differential pair of output in terms of PCIE_RXP and PCIE_RXN is given to the device controller. The blocks present in the proposed architecture works at different clock frequencies. As there is no in build clock source present in the FPGA, so to provide clock signals at different frequencies, Phase Locked Loop (PLL) is used. We have used the hard PLL here, which takes a clock signal at a particular frequency as input and provides clock signals at different frequencies as its output. The input frequency to the PLL is 100 MHz and the output frequencies are 100 MHz, 125 MHz, and 200 MHz.

3.2 SIMULATION RESULTS

3.2.1 SIMULATION RESULT OF FIFO_RX AND FIFO_TX

Different clock signals are used here for FIFO read and write so that it can support different port widths. The simulation result for FIFO_RX and FIFO_TX are shown in Fig. 3.2 and Fig. 3.3 respectively. From Fig. 3.2, it is clear that FIFO_RX takes 64-bit parallel input and gives 32-bit parallel outputs in two clock cycles. Similarly from Fig. 3.3, the FIFO_TX takes two 32-bits inputs and gives an output of 64 bits.

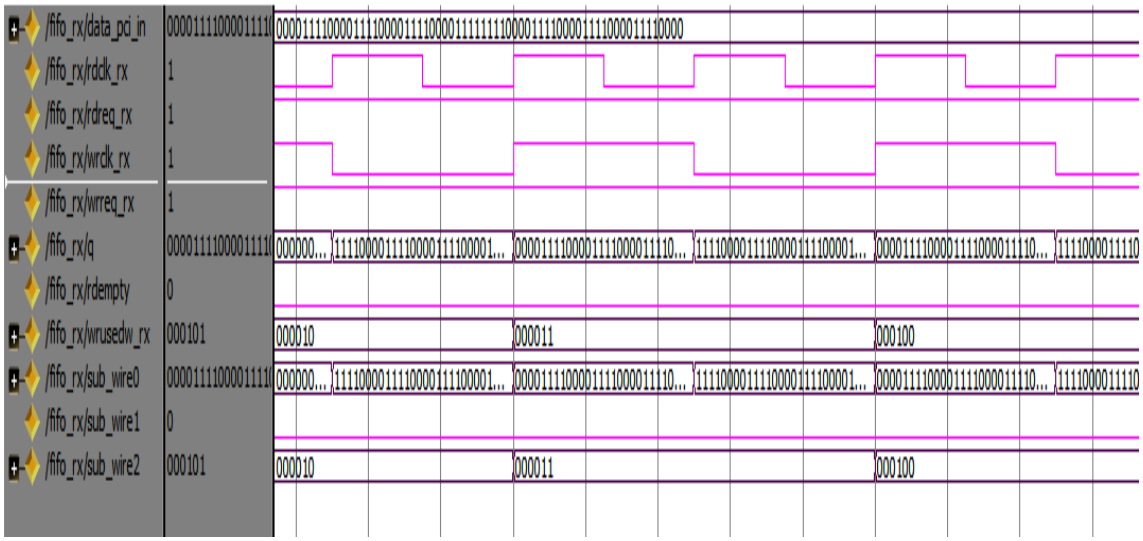

Fig. 3.2 Simulation result of FIFO_RX

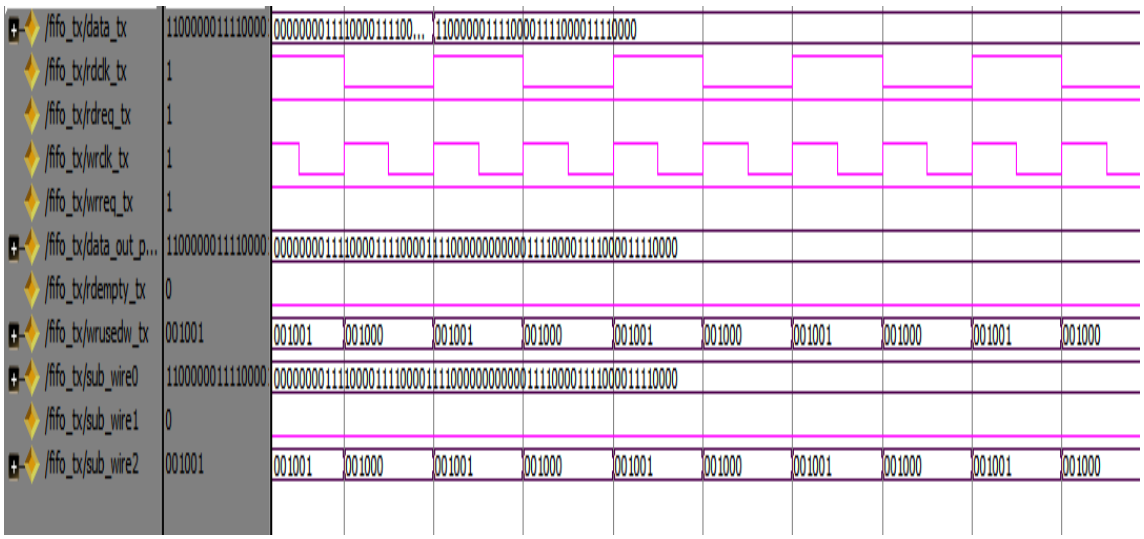

Fig. 3.3 Simulation result of FIFO_TX

3.2.2 SIMULATION RESULT OF GPIF CONTROLLER

The simulation result for GPIF controller is shown in Fig. 3.4. The results are verified with the proposed functionality of GPIF controller as discussed in the proposed design. The output of PLL is shown in Fig. 3.5. A delay of 10.7 ns is experienced while obtaining the output. For the above mentioned period, the output remains undefined.

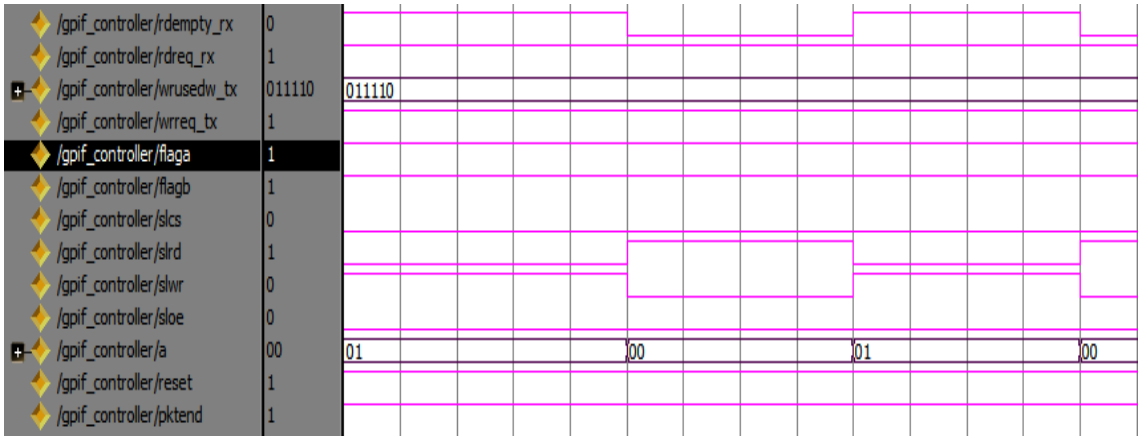

Fig. 3.4 Simulation result of GPIF controller

3.2.3 SIMULATION RESULT OF PLL

Here a clock frequency of 100 MHz is given as an input to the PLL and after a delay of 10.7 ns gets an output of 125 MHz, 50 MHz, 200 MHz and 100 MHz after the lock phase of the PLL.

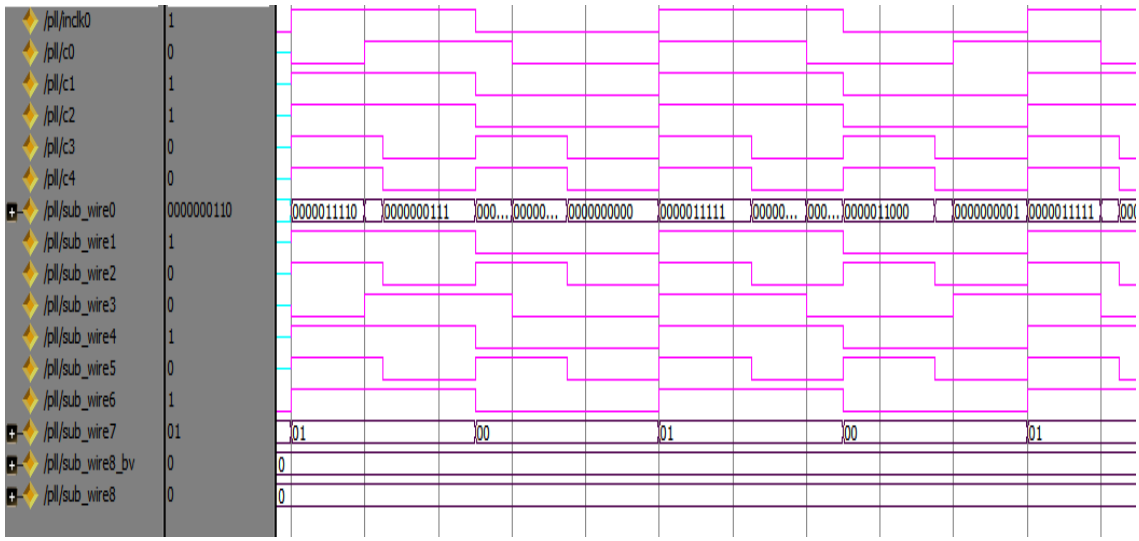

Fig. 3.5 Simulation result of PLL

3.2.4 SIMULATION RESULT FOR DATA TRANSFER

For the verification of data transfer between host and device, a 64-bit data has been given to FIFO_RX. As output, we got a 32-bit data. The transfer of data is being controlled by GPIF controller and is also passed through a buffer. The 32-bit data has been successfully transferred from the device to FIFO_TX. The above operation is controlled by GPIF controller. The simulated outputs are shown in Fig. 3.6 and Fig. 3.7 respectively.

Fig. 3.6 Simulation result showing data transfer between host controller and device controller

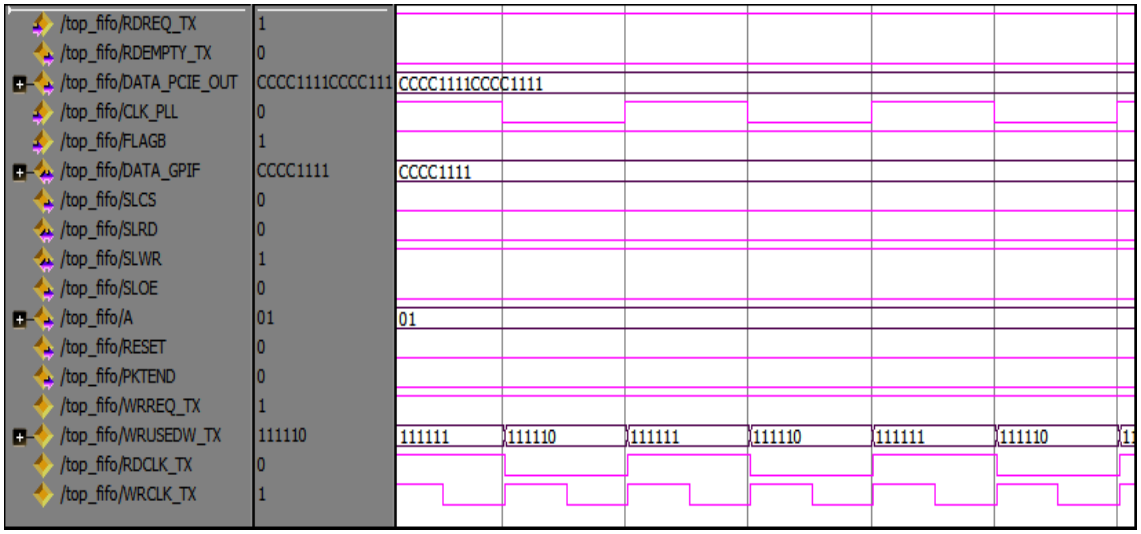

Fig. 3.7 Simulation result showing data transfer between device controller and host controller

3.2.5 Device utilization summary

The device utilization summary for the above designs on Altera's Stratix IV FPGA is shown in Fig. 3.8.

Top-level Entity Name	TOP_FIFO
Family	Stratix IV
Device	EP4SGX70DF29C3
Timing Models	Final
Logic utilization	< 1 %
Combinational ALUTs	183 / 58,080 (< 1 %)
Memory ALUTs	0 / 29,040 (0 %)
Dedicated logic registers	151 / 58,080 (< 1 %)
Total registers	151
Total pins	180 / 412 (44 %)
Total virtual pins	0
Total block memory bits	6,144 / 6,617,088 (< 1 %)
DSP block 18-bit elements	0 / 384 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	1 / 3 (33 %)
Total DLLs	0 / 4 (0 %)

Fig. 3.8 Device utilization summary for the proposed architecture

CHAPTER 4 Physical Layer Implementation

DESCRIPTION

SIMULATION RESULTS

4 PHYSICAL LAYER IMPLEMENTATION OF USB 3.0

4.1 DESCRIPTION

As discussed earlier the physical layer consists of two parts: transmitter and receiver. The transmitter part consists of scrambler, 8 bit to 10 bit encoder, parallel to serial converter, and transmitter differential driver. In this work, the first three parts of the transmitter have been implemented. Similarly from the receiver part, only serial to parallel converter, 8 bit to 10 bit decoder, and descrambler have been implemented. The functionality of the above blocks of transmitter and receiver of physical layer has been tested by writing code in ModelSim-Altera 6.6d. The FPGA used here is Altera's Cyclone II FPGA. The simulation result for transmitter and receiver are shown in Fig. 4.5 and Fig. 4.6 respectively.

4.2 SIMULATION RESULTS

4.2.1 SIMULATION RESULT OF SCRAMBLER

In this module an input of 8-bit data is given to the scrambler and it gives an 8-bit output using linear feedback register function. When the reset pin is '0', then it will give an output otherwise there will be no output.

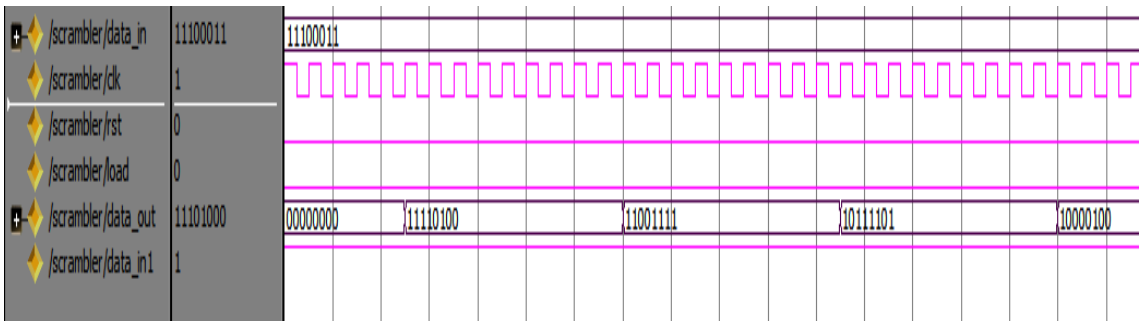

Fig. 4.1 Simulation result of Scrambler

4.2.2 SIMULATION RESULT FOR 8B/10B ENCODER

For this module an 8-bit data is given to this and depending on the control bit (ki), it gives the output. When the control bit is '0', it will give encode the data and gives an output of 10-bit, else all the bits of output will be zero.

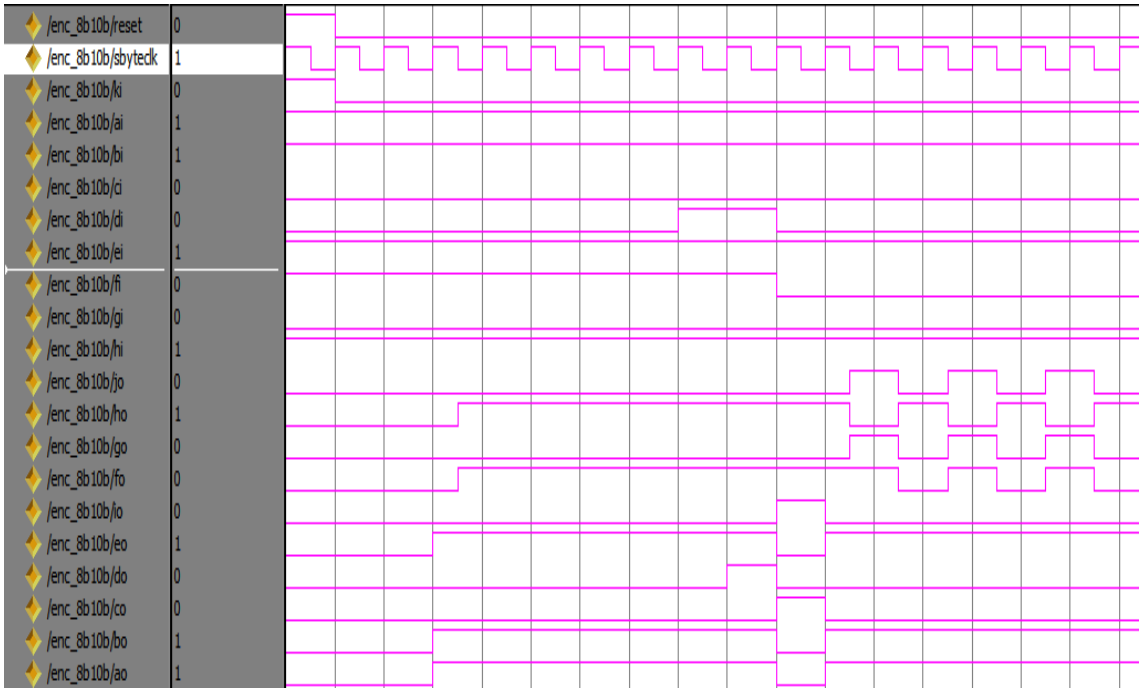

Fig. 4.2 Simulation result of 8b/10b encoder

4.2.3 SIMULATION RESULT FOR DESCRAMBLER

The functionality of the descrambler is similar to the functionality of scrambler and it also uses the same liner feedback shift register function for its operation.

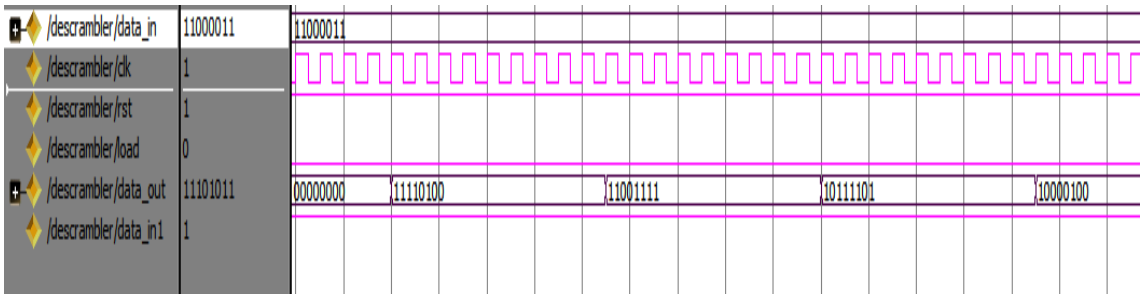

Fig. 4.3 Simulation result of Descrambler

4.2.4 SIMULATION RESULT FOR 8b/10bDECODER

Based on the reset input value decoder will decode the input data and gives an output of 8-bit. This 8-bit output may be a data word or control word. This will be defined by the control output bit (ko). If this bit is '1' the output will be a control word else data word.

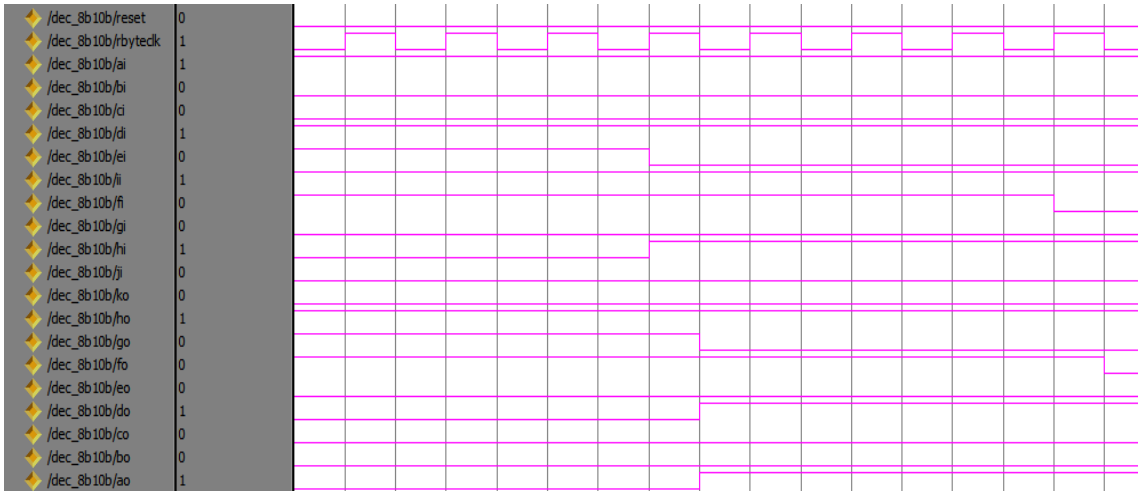

Fig. 4.4 Simulation result of 8b/10bdecoder

4.2.5 SIMULATION RESULT SHOWING 8-BIT DATA INPUT AND 1-BIT DATA OUTPUT

This is the final output of transmitter module which is implemented. Here after giving an input of 8-bit, we will get an output of 1-bit.

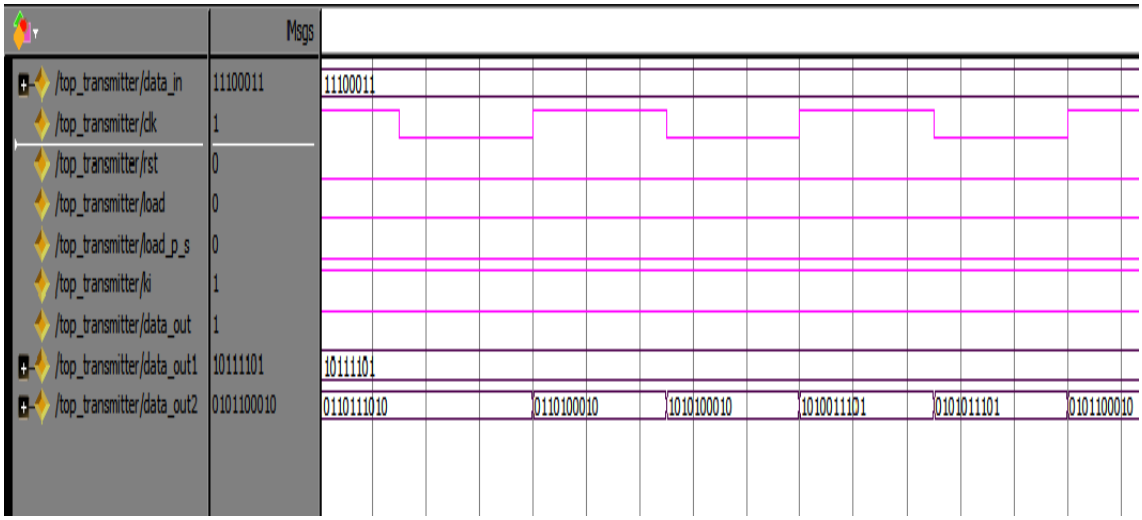

Fig. 4.5 Simulation result showing 8-bit data input and 1-bit data output

4.2.6 SIMULATION RESULT SHOWING 1-BIT DATA INPUT AND 8-BIT DATA OUTPUT

This is the final receiver module where an input of 1bit is given and results an output of 8-bit after descrambler and 8b/10bdecoder.

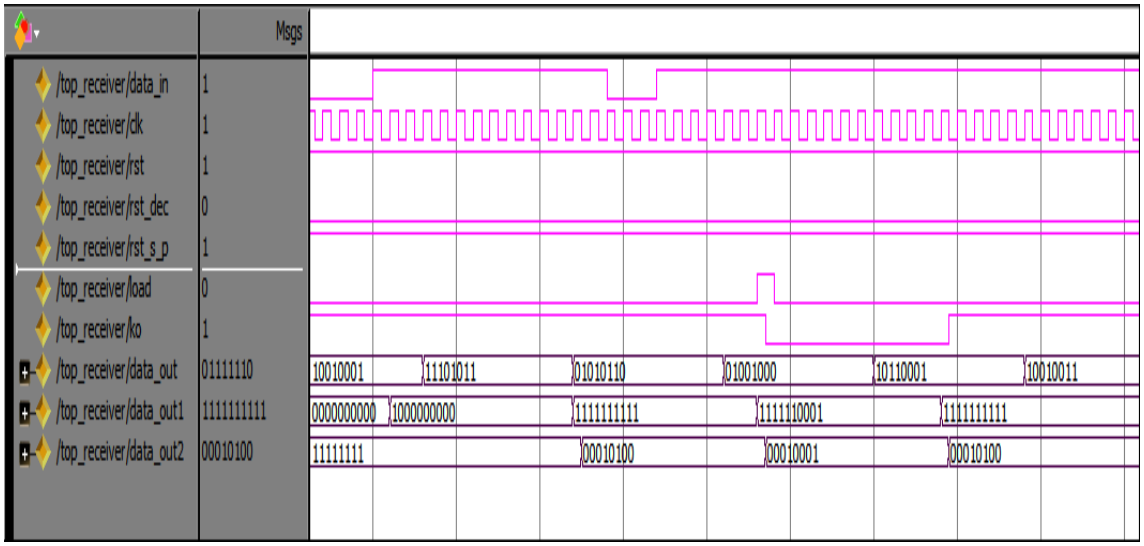

Fig. 4.6 Simulation result showing 1-bit data input and 10-bit data output

4.2.7 Device utilization summary

The device utilization summary for the above designs on Altera's Cyclone II FPGA is shown in Fig. 4.7 and Fig. 4.8.

Revision Name	transmitter
Top-level Entity Name	top_transmitter
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	88 / 18,752 (< 1 %)
Total combinational functions	73 / 18,752 (< 1 %)
Dedicated logic registers	62 / 18,752 (< 1 %)
Total registers	62
Total pins	14 / 315 (4 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Fig. 4.7 Device utilization summary for transmitter block

Revision Name	top_receiver
Top-level Entity Name	top_receiver
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	104 / 18,752 (< 1 %)
Total combinational functions	90 / 18,752 (< 1 %)
Dedicated logic registers	53 / 18,752 (< 1 %)
Total registers	53
Total pins	15 / 315 (5 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Fig. 4.8 Device utilization summary for receiver block

CHAPTER 5 CONCLUSION

5 CONCLUSION

The individual blocks in the proposed architecture has been tested and verified successfully on ModelSim - Altera. The simulation results have been shown. We have used here the hard IP of Stratix-IV FPGA. Phase locked loop, present in Stratix-IV FPGA has been used to provide clock signal at different frequencies. Individual blocks have been given clock signal at different frequencies for maintaining synchronization. FIFO has been used to convert 64-bit output of PCIe hard-IP into 32-bit to make it compatible with GPIF controller. Finally, data has been successfully transferred from the host controller to the device controller.

The SuperSpeed data communication for USB 3.0 has been achieved by implementing the proposed architecture. The up-conversion and down-conversion between 64-bit and 32-bit bus is found to work efficiently. Simulated results of data transmission between host controller and device controller show the feasibility of the proposed architecture. The functionality of physical layer of USB 3.0 is simulated based on the defined architecture and the results are found to be satisfactory. All the simulations are done on ModelSim-Altera 6.6d platform using VHDL.

REFERENCE

[1] Dr. Andreas C. Wolf, Dr. Wolf Wireless GmbH, Dr. Christoph Scheytt, "15 Gbps Communication over an USB3.0 Cable and Even More," 2012 9th International Multi-Conference on *Systems, Signals and Devices (SSD)*, Mar. 2012, pp. 1 – 3.

[2] Roberto Ammendolaab, Andrea Biagionic, Giacomo Chiodic, etal, "High Speed Data Transfer with FPGAs and QSFP+ Modules," *2010 IEEE Nuclear Science Symposium Conference Record (NSS/MIC),* Oct. 30 2010-Nov. 6 2010, pp. 1323 – 1325.

[3] Lin, M.-S., Tsai, C.-C., Chang, C.-H., Peng, Y.-C., Yu, T.-H., Chien, J.-Y., et al. (2009). A 5Gb/s low-power PCI express/USB3.0 ready PHY in 40nm CMOS technology with high-jitter immunity. *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, (pp. 177-180).

[4] Hu Li, Yuan'an Liu, Dongming Yuan, Hefei Hu, "A Wrapper of PCI Express with FIFO Interfaces based on FPGA," *2012 International Conference on Industrial Control and Electronics Engineering*,

[5] Hossein Kavianipour, Steffen Muschter and Christian Bohm, " High Performance FPGA-based DMA Interface for PCIe," *2012 18th IEEE-NPSS Real Time Conference (RT)*, June 2012, pp. 1 – 3.

[6] Universal Serial Bus 3.0 Specification Revision 1.0: http://www.usb.org.

[7] Universal Serial Bus 3.0 Specification Revision 1.0: http://www.usb.org.

[8] PIPE Architecture 3.0: http://www.pci-sig.com.

[9] Aug. 2012, pp. 525 – 529.Ranianand Venkata, Wilson Won, etal, "Architecture and Methodology of a SoPC with 3.25Gbps CDR based Serdes and Gbps Dynamic Phase Alignment," *IEEE 2003 CUSTOM INTEGRATED CIRCUITS CONFERENCE*, Sept. 2003, pp. 659 – 662.

[10] STRATIX IV Device hand book: Volume 2: http://www.altera.com.

[11] Shao-Hang Hung, Chih-Feng Chao, Yu-Chun Yan, etal., "Independent Component Analysis Hard-IP integration System on Programmable Chip (SOPC) Platform," *TENCON 2010 - 2010 IEEE Region 10 Conference*, Nov. 2010, pp. 1705 – 1709.

[12] Edin Kadric, Naraig Manjikian, Zeljko Zilic, "AN FPGA IMPLEMENTATION FOR A HIGH-SPEED OPTICAL LINK WITH A PCIE INTERFACE," *2012 IEEE International SOC Conference (SOCC),* 12-14 Sept. 2012, pp. 83 – 87.

[13] IP compiler for PCIe user guide: http://www.altera.com.

[14] *PCI Express hard IP*. (n.d.). Retrieved from www.altera.com: http://www.altera.com/technology/high_speed/protocols/pcie-hard-ip/pro-hard-ip.html.

[15] Wu, Y., & Qiu, Y. (2012). The 8-bit Parallel CRC-32 Research and Implementation in USB 3.0. *Computer Science Service System (CSSS), 2012 International Conference on*, (pp. 1079-1082).

[16] *Transceiver Overview: Stratix IV and HardCopy IV*. (n.d.). Retrieved from www.altera.com: http://www.altera.com/devices/fpga/stratix-fpgas/stratix-iv/transceivers/stxiv-transceivers.html.

[17] Application note TUSB7320: http://www.ti.com.

[18] Application note CYUSB3014: http://www.cypress.com.

[19] Yao, W., Xuan, Z., Lei, T., & Jingcheng, L. (2010). A Date Transfer and Control System Solution Based on EZ-USB 2.0 for FPGA Applications. *E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on*, (pp. 1-4).

[20] Ranianand Venkata, Wilson Won, etal, "Architecture and Methodology of a SoPC with 3.25Gbps CDR based Serdes and Gbps Dynamic Phase Alignment," *IEEE 2003 CUSTOM INTEGRATED CIRCUITS CONFERENCE*, Sept. 2003, pp. 659 – 662.

[21] *Linear feedback shift register*. (n.d.). Retrieved from www.wikipedia.org: http://en.wikipedia.org/wiki/Linear_feedback_shift_register.